WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term:

Display: 10 Documents in Display Format: TI Starting with Number 1

Generate: ○ Hit List ◉ Hit Count ○ Side by Side ○ Image

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences | Cases

Search History

DATE: Sunday, February 09, 2003 Printable Copy Create Case

Set Name side by side	Query	Hit Count	Set Name result set
DB=USPT; PLUR=YES; OP=OR			
L3	L2 and 4679038.uref.	13	L3
L2	L1 and 345.clas.	1336	L2
L1	memory and (section or band or strip) and compression	27802	L1

END OF SEARCH HISTORY

WEST

Generate Collection | Print

Search Results - Record(s) 1 through 10 of 13 returned.

☐ 1. Document ID: US 6108014 A

L3: Entry 1 of 13 File: USPT Aug 22, 2000

US-PAT-NO: 6108014
DOCUMENT-IDENTIFIER: US 6108014 A

TITLE: System and method for simultaneously displaying a plurality of video data objects having a different bit per pixel formats

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

☐ 2. Document ID: US 6067098 A

L3: Entry 2 of 13 File: USPT May 23, 2000

US-PAT-NO: 6067098
DOCUMENT-IDENTIFIER: US 6067098 A

TITLE: Video/graphics controller which performs pointer-based display list video refresh operation

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

☐ 3. Document ID: US 6020894 A

L3: Entry 3 of 13 File: USPT Feb 1, 2000

US-PAT-NO: 6020894
DOCUMENT-IDENTIFIER: US 6020894 A

TITLE: Full-color desktop publishing system

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Image

☐ 4. Document ID: US 6002411 A

L3: Entry 4 of 13 File: USPT Dec 14, 1999

US-PAT-NO: 6002411
DOCUMENT-IDENTIFIER: US 6002411 A

TITLE: Integrated video and memory controller with data processing and graphical processing capabilities

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC
Draw. Desc | Image

☐ 5. Document ID: US 5995120 A

L3: Entry 5 of 13 File: USPT Nov 30, 1999

US-PAT-NO: 5995120
DOCUMENT-IDENTIFIER: US 5995120 A

TITLE: Graphics system including a virtual frame buffer which stores video/pixel data in a plurality of memory areas

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC
Draw. Desc | Image

☐ 6. Document ID: US 5801716 A

L3: Entry 6 of 13 File: USPT Sep 1, 1998

US-PAT-NO: 5801716
DOCUMENT-IDENTIFIER: US 5801716 A

TITLE: Pipeline structures for full-color computer graphics

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC
Draw. Desc | Image

☐ 7. Document ID: US 5754750 A

L3: Entry 7 of 13 File: USPT May 19, 1998

US-PAT-NO: 5754750
DOCUMENT-IDENTIFIER: US 5754750 A

TITLE: Method and apparatus for displaying a page with graphics information on a continuous synchronous raster output device

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

☐ 8. Document ID: US 5748986 A

L3: Entry 8 of 13 File: USPT May 5, 1998

US-PAT-NO: 5748986
DOCUMENT-IDENTIFIER: US 5748986 A

TITLE: Method and apparatus for displaying a page with graphics information on a continuous synchronous raster output device

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

☐ 9. Document ID: US 5509115 A

L3: Entry 9 of 13 File: USPT Apr 16, 1996

US-PAT-NO: 5509115
DOCUMENT-IDENTIFIER: US 5509115 A

TITLE: Method and apparatus for displaying a page with graphics information on a continuous synchronous raster output device

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

☐ 10. Document ID: US 5502804 A

L3: Entry 10 of 13 File: USPT Mar 26, 1996

US-PAT-NO: 5502804
DOCUMENT-IDENTIFIER: US 5502804 A

TITLE: Method and apparatus for displaying a page with graphics information on a continuous synchronous raster output device

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | KWIC
Draw. Desc | Image

Generate Collection | Print

Terms	Documents
L2 and 4679038.uref.	13

Display Format: TI Change Format

L3: Entry 9 of 13 File: USPT Apr 16, 1996

DOCUMENT-IDENTIFIER: US 5509115 A
TITLE: Method and apparatus for displaying a page with graphics information on a continuous synchronous raster output device

Abstract Text (1):
An apparatus and method used to perform image rendering is capable of producing complex, high resolution images for a continuous synchronous raster image output device, such as a laser printer or a video display, using a minimum of random access memory. High level graphics instructions defining an image to be displayed or printed are provided by a processor executing an applications program. These instructions are interpreted to generate a set of graphics orders that are subsequently processed to create a bitmap image for output to a display or printer device. The graphics order processing system is independent of the central processor and thereby significantly reduces the central processor's image output overhead.

Brief Summary Text (2):
The present invention is an apparatus and method used to perform image rendering in a manner which is capable of producing complex, high resolution images for a continuous synchronous raster image output device, such as a laser printer or a video display, using a minimum of random access memory. This invention yields substantial performance improvements for the processor to which the output device is attached.

Brief Summary Text (3):
A continuous synchronous raster image output device requires delivery of the output data in a fixed period of time. If the data is not delivered within the fixed period of time, the output image is corrupted. The invented method and apparatus decomposes basic graphics functions into a compact series of orders or commands suitable for realtime processing and then generates the output image on the fly. By way of contrast, excepting for applications with limited or no graphics capability, prior art techniques for continuous synchronous raster image output devices require that an entire image be constructed in memory before the image is sent to the output device.

Brief Summary Text (4):
The invented technique defines the graphic content of an image in much less memory than the rendered image would otherwise require, but in a form that can be processed and delivered to an output device at the speed required by the output device. In order to accomplish this, graphics functions are represented as low level primitives, referred to herein as orders, to guarantee that they can be converted to bitmapped images in realtime. Additionally, the quantity of orders or commands is reduced by eliminating redundant or overlapping graphics functions to minimize the memory used by the orders. This approach substantially reduces the output-related processing overhead of the central processor since it is only required to generate a high level language definition of the output image.

Brief Summary Text (5):
For printer applications, the invention uses a realtime output processor which operates on a small section of a page image at a time. Such a section is called a band. A band consists of a predetermined number of scanlines where each scanline is a series of pixels to be delivered to the output device at the speed required by the output device.

Brief Summary Text (6):
After building a first band, a second band is created while the first band is being printed. Using prior art techniques, and given the present speed of processors used to generate bitmaps of the image to be printed which is sent to the output device, it would not be possible to finish creating a bitmap for the second band before the printer finished printing the first band. That is, since continuous synchronous raster image printers print at a fixed speed, to ensure that a page is printed correctly, prior art techniques require the creation of a bitmap image of the entire page to be printed before any part of the image is passed to the printer.

Brief Summary Text (7):
By using a technique for generating bitmap images which are less than the complete page, and sending only that portion to the printer, while the bitmap image for another portion of the page is being created, the present invention results in a great cost savings by using less memory. Additionally, even if there is sufficient memory to build a complete page, by using the banding technique of the present invention, a second page can be constructed in one band while a first page in another band is being printed, thereby increasing the throughput of the printer.

Brief Summary Text (8):
The present invention accomplishes its goal by deferring the execution of application generated commands that draw pixels in the page image. These commands are converted to graphics orders rather than directly to bitmaps. The graphics orders take up much less memory than corresponding bitmaps, but can be converted to bitmaps for sending to a printer at a speed fast enough to keep up with the printer.

Brief Summary Text (9):
Since a single graphics command may affect more than one band, an order may be processed multiple times. In this connection, order command blocks, which are represented as tabular structures in memory, include information to allow selection of those graphics commands needed for a given band.

Brief Summary Text (10):
More particularly, image rendering according to the present invention uses an order construction step and two image drawing steps. The order construction step is used to build orders for deferred execution. Applications commands for pixel drawing in memory other than the page image result in bitmaps stored in user memory and do not result in stored orders because these commands are processed immediately. The bitmaps produced by immediate memory block transfers (blits) later may be used by a PDL interpreter as source bitmaps for blits into the page image.

Brief Summary Text (11):
Inputs to the order construction step are presented through a graphics subsystem interface such that each graphics call delivers arguments which the order construction step reduces to a compact command block which may include a "band" number to be used by the image drawing steps to select a subset of commands for each band of the page to be drawn.

Brief Summary Text (12):
The order construction step attempts to eliminate redundant orders by combining multiple inputs into single command blocks thus reducing the number of command blocks and the memory required to contain them.

Brief Summary Text (13):
The first (immediate) image drawing step proceeds in parallel with the generation of orders that draw objects in the page image. Objects drawn in this step are stored in the above-mentioned user memory for use as source data for the second drawing step, but the order command blocks for them are released for reuse.

Brief Summary Text (14):
The second (deferred) image drawing step starts when all orders for the page image have been generated. This step may process order commands multiple times, once for each image band affected. Orders that do not affect the current band are skipped.

Brief Summary Text (15):
The first band affected by an order command is identified by the band number included in the command. When an order command is processed, the band number is updated to the next band number if the order command affects multiple bands.

Brief Summary Text (16):
The invented technique is applicable not only to printers, but to raster scanned displays as well. The ever increasing x-y resolution of computer displays, in combination with the even more rapidly increasing color (z) resolution, has resulted in video memory requirements that are comparable to those of high resolution graphics printers. For example, a 600 dpi laser printer requires approximately 4MB of RAM, whereas high resolution displays now on the market require approximately 3MB of RAM. However, there have not been corresponding improvements in the CPU to video controller interface. Consequently, CPU performance suffers due to the large number of CPU cycles required to update the video RAM. Even though CPU throughput has increased dramatically, such increased performance is not realized during video transfers since the CPU must wait to carry out each memory transaction to the video memory across a relatively slow interface.

Brief Summary Text (17):
The video display update bottleneck is being addressed through two implementations: execution of higher level drawing operations within the display adapter itself and placement of the video adapter memory on a bus more tightly coupled to the microprocessor.

Brief Summary Text (19):
The move to place the display adapter closer to the microprocessor on its "local" bus requires a significant change to current PC motherboard architecture. Two proposals to modify this architecture are underway by the Video Equipment Standards Association (VESA) and Intel Corporation through the Peripheral Component Interconnection (PCI) bus. Should either of these standards be adopted, video screen updates will significantly improve due to the removal of the current interface bottleneck present in current architectures. However, overall system performance will be less than optimal as the processor must still carry out all operations to the display memory. The proposed local bus standards simply increase the efficiencies of accessing this memory.

Detailed Description Text (7):
For example, in the Postscript instructions above, the newpath operator causes the PDL interpreter to initialize internal information to control subsequent operations. The moveto operator establishes an initial position within the coordinate system. The lineto operator establishes a new position with an unexpressed or "uninked" line between the initial position and the new position. When the interpreter processes the stroke operator, it will generate one or more graphics commands to underlying pixel drawing routines which draw the line in the page image memory.

Detailed Description Text (8):
In this case, the graphics calls are handled by the output controller synchronously. That is, a particular graphics call generates a portion of an image which is to be printed which is stored in a page buffer after which control is returned to the PDL interpreter. The interpretation and drawing steps continue work on the same page image until a page termination operator is found. Then the page image memory is copied to the output device by the hardware. The output copying step may operate asynchronously to the interpreter and graphics functions if there is enough memory to allow the interpreter to produce an image for the next page while the hardware delivers output from the previous page via DMA or other means.

Detailed Description Text (26):
Generally each of the six classes has at least two calls, one for writing to the page image bitmap which the graphics subsystem interface manages; the other for a PDL interpreter managed destination, typically, a RAM user memory.

Detailed Description Text (28):
Generally, bitmap image drawing calls expect user and halftone bitmap descriptors as

input. A user bitmap descriptor contains bitmap origin byte address, i.e., the memory address of the upper left corner of the bitmap, bitmap width in pixels, i.e., the number of pixels per scanline, and bitmap height in pixels, i.e., the number of scanlines. Bitmap width and height may be arbitrary pixel values. Clipping occurs to the specified bitmap width and height for destination bitmaps; no source bitmap clipping is performed.

Detailed Description Text (31):
This feature must be used sparingly and only where truly needed due to the expense in memory and processor time. It is not intended as a substitute for PDL interpreter management of required memory. When the PDL interpreter and the other portions of output controller 15 operate synchronously, no problem arises. If the two processes run asynchronously as they do when the controller includes hardware assistance or when double buffering is in effect, the output controller must make a copy of a "volatile" bitmap so that the bitmap does not change before it can be used. The caller, i.e., the PDL interpreter can flag a bitmap as volatile in its descriptor.

Detailed Description Text (33):
A functional block diagram of output controller 15 and bit map image generator 17 as implemented in the prior art will now be described with reference to FIG. 3. Specifically, output controller 15 and bitmap image generator 17 comprise a graphics interface subsystem 23 which receives the graphics calls generated by PDL interpreter 21 based upon the PDL instructions 13 generated by application 11. The graphics interface subsystem passes the graphics calls to immediate blit processor 29. User memory 24 is a random access memory used by PDL interpreter 21 as a temporary storage of source bitmaps which will be passed to graphics interface subsystem 23 as source arguments to graphics calls. Memory manager 43 allocates and deallocates RAM memory blocks as requested by graphics interface subsystem 23. When an End Page instruction is called by the PDL interpreter, the actual bitmap images to be printed are transferred from page buffer 42 to DMA hardware or a FIFO buffer 53 for transfer to a printer by operation of output interface 47, output interrupt handler 49, and graphics interface subsystem 23.

Detailed Description Text (36):
When the PDL interpreter calls the graphics interface subsystem 23 to start a new page, the graphics interface subsystem calls memory manager 43 to allocate memory for the page image.

Detailed Description Text (41):
Interrupt handler 49 operates by continuing the delivery of data to the DMA or FIFO 53 and by starting a queued image, if any, when the output for the current image is complete. When the output of a page image is completed, interrupt handler 49 calls a completion routine in graphics interface subsystem 23 to announce this completion. At this time graphics interface subsystem 23 can return the page image buffer 42 to memory manager 43 or re-use it for another page.

Detailed Description Text (42):
FIG. 4 is a functional block diagram showing the functions of output controller 15 and bitmap image generator 17 when implemented according to the teachings of the present invention. Before describing the elements of FIG. 4 in detail, it will be helpful to explain the major differences between the technique for rendering the output image in small bands according to the present invention and rendering it in a full page image buffer according to the prior art. These differences are as follows:

Detailed Description Text (49):
4. A capability for updating the stored orders by the realtime blit processor to eliminate those that it completes or mark those that it must reprocess in a way so it can efficiently reprocess an order that spans multiple bands.

Detailed Description Text (50):
While the invented technique may be implemented in software, when implemented using co-processor hardware, the hardware replaces both the immediate blit processor and the realtime blit processor. In addition, in the preferred embodiment, a hardware implementation incorporates the output FIFO or DMA device 53. A hardware

implementation of the realtime blit processor does the same type of order updating as that performed by a software implementation. However, since a hardware implementation runs much faster than a software implementation, the hardware implementation can band much more complex pages and keep up with much faster output devices. It also may employ a separate hardware bus for access to the band buffers 41a, 41b and 41c.

Detailed Description Text (54):
PDL interpreter 21 calls graphics interface subsystem 23 which is a set of standard interface routines for graphics services. The graphics interface determines whether it can perform an immediate blit based on whether or not the destination is user memory 24 or band buffers 41. If it cannot do an immediate blit, it calls order construction function 31 to create stored orders 35.

Detailed Description Text (57):
This function draws pixels in user memory 24 as requested and returns. This function is known in the prior art and is performed whenever pixel drawing commands affect a user memory destination.

Detailed Description Text (59):
Order construction processing 31 generates orders 35 which are stored in a memory for subsequent use during realtime blit processing performed by realtime blit processor 37. The orders include initialization, program flow control, bit-block transfer (for rectangular regions of bitmaps) and scanline transfers (for arbitrarily-shaped regions of bitmaps). Blit processor 37 executes both bit-block and scanline transfers.

Detailed Description Text (62):
This function keeps records of the orders that may be compressed until informed by the order construction processing function that it must wrap up a set of orders. Then it reinitializes its internal information to start collecting information about the next set. This function is very important in cases where many small scanline blit orders operate on a small region of the page image. Such compaction provides major reductions in the amount of memory required in cases because the blits overlap.

Detailed Description Text (66):
When the PDL interpreter issues a call to say it has completed all the requests for a page, graphics interface subsystem 23 starts the realtime processing phase. It traverses stored orders 35 once for each band to be output. When all orders that effect a band have been processed, it calls output interface 47 to queue the band for output or start it immediately if the DMA or FIFO buffer output hardware 53 is available. If another band buffer 41a, 41b or 41c is available, it will construct the next band. When no more buffers are available, it returns to the graphics interface subsystem 23 which returns to PDL interpreter 21. The PDL interpreter may continue interpreting input and working on the next page. From this point on, realtime blit processor 37 is driven by interrupts initiated by output hardware 53.

Detailed Description Text (67):
Realtime blit processor 37 converts the stored orders into a bitmap image corresponding to a band or section of the page to be printed which is stored in band buffers 41a, 41b or 41c.

Detailed Description Text (68):
Specifically, interrupt handler 49 is notified by output hardware 53 when a band has been delivered to the printer. Realtime blit processor 37 is then called as an "extended interrupt thread" to continue its processing.

Detailed Description Text (69):
A "band fault" is declared when a stored order pertains to a bitmap image that crosses a band boundary. In such event, the stored order is marked and updated so that the order will be more efficiently processed in the next band. For example, if a character image straddles the boundary between band n and band n+1, the stored order for creating the bitmap image of the character is first processed in band n. When the band boundary is reached, the order is marked and updated with a modified

source address, destination address and height for the character image so that only the remaining portion of the image is retrieved from memory when the order is processed in band n+1.

Detailed Description Text (70):
Band Buffers 41a, 41b, 41c

Detailed Description Text (71):
Band buffers 41a, 41b and 41c is a random access memory used to hold the bitmapped page image. If banding is not used, a single band buffer is used to hold a complete page. Otherwise there may be 2 or 3 band buffers of a size much smaller than the complete page.

Detailed Description Text (72):
Memory Manager 43

Detailed Description Text (73):
Memory manager 43 allocates and deallocates blocks of memory in response to calls from graphics interface subsystem 23, order construction 31, order compaction 33 or realtime blit processor 37. Memory manager 43 is a software function that manages the memory containing the stored orders and the band buffers according to well known memory management techniques which need to be employed when the other software components need to acquire or free blocks of memory. The actual bitmap images to be printed are transferred from the band buffers 41 to DMA hardware or a FIFO buffer 53 for transfer to a printer by operation of output interface 47, output interrupt handler 49, realtime Blit processing 37 and memory manager 43. In this connection, although three band buffers are shown in FIG. 4, in practice, there may be two, three or more band buffers, although in most cases two or three band buffers is sufficient.

Detailed Description Text (75):
Output interface 47 receives completed bands from the realtime blit processor and either passes them on to the output hardware 53 or queues them for output interrupt handler 49 to pass to the output hardware when it is no longer busy. With the help of output interrupt handler 49, output interface 47 monitors synchronization signals from the printer engine to determine when it can deliver a band for output. Print engines send a "frame sync" signal to indicate top of page. The output interface cannot deliver any data until the "frame sync" signal is true.

Detailed Description Text (76):
When a FIFO buffer is used as the output hardware, the software moves data from the band buffer to the FIFO a scanline at a time. A DMA controller can be used the same way. Top and left margins are created by delivering zeroes for an output device that writes black or by delivering ones for an output device that writes white.

Detailed Description Text (78):
As the output hardware finishes delivering a band to a video interface of the print engine, it generates an input. The output interrupt handler gets the next band from its queue and passes it to the output hardware. It then returns the previously completed band buffer so the realtime blit processor may reuse it.

Detailed Description Text (79):
When the output interrupt handler returns, realtime blit processor 37 runs again until it has rendered another band before the graphics interface system returns to the PDL interpreter. This "extended interrupt thread" is implemented differently on different CPUs depending on their architectures. The "extended interrupt thread" runs on a separate stack with CPU priority lowered so that it can be interrupted.

Detailed Description Text (82):
In the preferred embodiment, the functional elements shown in FIG. 4 are implemented in hardware, although if one or more of such elements are implemented in software, the function performed by each such element is actually performed by operation of a microprocessor executing a set of instructions in a memory such that the microprocessor generates control signals appropriate to the function being performed. In this connection, just as the specific hardware implementation details

are not needed for a person skilled in the field of the invention to make and use the same, the particulars of a microprocessor, memory, data and address bus, clock and the like needed to implement the invention in software would be readily apparent to a person skilled in the art based upon the description provided herein. The following is a complete software specification for implementing the invention.

Detailed Description Text (94):
Banding involves building an image to be output in strips or "bands". Generally, strip n+1 is being composed while strip n is being output to the print engine.

Detailed Description Text (95):
Banding is a process in which the page to be printed is represented in an intermediate form which describes the graphics operations necessary to construct the bit mapped image for the page. This intermediate form is commonly referred to as a `display list`. The PDL or PCL emulator firmware running on the main processor generates this display list before the print engine begins the actual printing process. The Graphics Execution Unit in PBC executes the display list to form the bit mapped image in bands after the print engine is started.

Detailed Description Text (96):
Banding is desirable in printer controllers due to its potential memory savings and system performance improvements. For banding to be viable, the intermediate representation must meet two requirements: it must fit in considerably less memory than the full image would require, and the commands to generate a band must be executable within the time it takes to print the previous band.

Detailed Description Text (97):
The graphics orders of PBC are designed to be an extremely compact representation of the printed page. RLL compression of graphics data and the elimination of redundant information in the display list enables the intermediate form of the printed page to be significantly smaller than the actual bit mapped image.

Detailed Description Text (103):
Banding has not been possible in PDL laser printer applications due to the potential complexity of the images which can be constructed by these languages. An acceptable intermediate representation had not been developed until the introduction of the PBC graphic orders. Using the compact PBC orders, all PostScript pages, except the most complex, can be represented in less than 250 Kbytes of memory. All pages can be represented with additional memory.

Detailed Description Text (105):
A bitmap is a two dimensional array of memory bits, generally referred to as pixels or dots, as shown in FIG. 7. When stored in memory, this two dimensional array is represented in a contiguous run of bits with no gaps or holes.

Detailed Description Text (107):
The X dimension of the bitmap array is referred to as pitch, width, or warp. Width is commonly used to describe smaller arrays, such as character bitmaps. Warp is typically used when describing page image bitmaps. The warp of a bitmap is a particularly useful parameter in that it is the value that, when added to the position in memory of a particular pixel of a scanline, results in a position in the same pixel column in the next lower scanline; warp is the amount to move to result in a Y-only movement within the bitmap.

Detailed Description Text (114):
1.6 MEMORY ADDRESSING

Detailed Description Text (118):
The following sections describe the data structures used by the PBC to control its operation and render graphical images.

Detailed Description Text (120):
The PBC contains two independent subsystems to simultaneously render and print page images: the Graphics Execution Unit (GEU) and the Print Engine Video Controller (PVC). The Graphics Execution Unit is used to execute a display list of graphic

orders and render a page image, while the Print Engine Video Controller is used to transfer the rendered page image data to the laser print engine. This section describes the software interfaces to PBC for controlling the Graphics Execution Unit and Print Engine Video Controller.

Detailed Description Text (121):
The software interfaces to the Graphics Execution Unit and Print Engine Video Controller employ a combination of resources: PBC registers, memory-resident command blocks, and microprocessor interrupts. The following is a brief overview of the operation of the Graphics Execution Unit and Print Engine Video Controller.

Detailed Description Text (123):
A functional block diagram of the PBC's Graphics Execution Unit and Print Engine Video Controller is shown in FIG. 9. The Graphics Execution Unit is started by writing the beginning address of a display list to an PBC register. The Graphics Execution Unit executes the display list and renders a page or band image. When the end of the display list is reached, the Graphics Execution Unit generates an interrupt to the microprocessor, and waits for another display list address. A second display list address can be loaded while the Graphics Execution Unit is working on the first display list. In this case, upon completion of the first display list, an interrupt is generated and the second display begins executing immediately. A second interrupt is generated upon completion of the second display list.

Detailed Description Text (124):
The Graphics Execution Unit can render an entire page from a display list, or multiple bands from a single "banded" display list. A banded display list contains band information near the beginning of the list. This band information includes the address and size of the band as well as the band number that is to be processed. The Graphics Execution Unit uses this information to determine which orders from the display list to process and where in memory to create the bit map image for this specific band. Thus after one band is fully rendered and the Graphics Execution Unit generates an interrupt, software simply adjusts the display list to reflect the next band information and then restarts the Graphics Execution Unit.

Detailed Description Text (125):
This interface between the processor and the Graphics Execution Unit allows complete freedom in the design of the banded memory system. The quantity, size, and location of band buffers is determined by the main processor's software. These band parameters can be dynamically altered to suit a particular application or page complexity. The desired band number is included to allow for non-sequential band processing applications such as duplex printing.

Detailed Description Text (126):
After a page image (or band image) is created by the Graphics Execution Unit, the Print Engine Video Controller is used to transmit the page image to the print engine. The Print Engine Video Controller is started by writing an address of a print command block, or PCB to an PBC register. Upon activation, the Print Engine Video Controller reads the parameters from the PCB, fetches the first page image data from memory, and then waits for the beginning of page signal (Frame sync, FSYNC) from the print engine. When FSYNC is detected, the Print Engine Video Controller generates a "band begin" interrupt, and starts transmitting the page image video data to the print engine. When the entire page or band image has been transmitted, a "page end" interrupt is generated, allowing software to reuse the page image memory space.

Detailed Description Text (127):
When banding, the Print Engine Video Controller must be given several PCB addresses throughout the course of one page. After each "band begin" interrupt is generated, software may load another PCB address to the next band's page image. It may reuse the band image memory and the PCB structure used in the previous band at this time.

Detailed Description Text (128):
The Print Engine Video Controller automatically manages all parameters associated with the video image. Top margins and left margins are automatically inserted by the

Video Controller. The Page Width parameters are used to calculate the beginning address of each scanline. The page or band height parameters are used to determine the end of the page or band.

Detailed Description Text (129):
This type of print engine video interface eliminates all software overhead associated with the transfer of the bit mapped image to the print engine. Software merely starts the transfer and waits for the end of page or band interrupt. This frees the software to start processing the next page.

Detailed Description Text (130):
This type of print engine interface also saves a significant amount of hardware cost. Static RAM's or FIFO's are not necessary as the bit mapped image is DMA'ed directly from the DRAM memory. The Print Engine Video Controller uses an efficient page mode access to DRAM to maximize memory bandwidth and contains an internal 4 word queue for printer video data.

Detailed Description Text (135):
The Display List Address Register in PBC is "double buffered" to maximize Graphics Order execution. A second operation may be started prior to the completion of the first operation. The second operation will be automatically started upon completion of the first operation. The Status bit DLA EMPTY indicates when a subsequent display list starting address may be loaded. Refer to Section 2.2.7 for a detailed discussion of the function of this status bit.

Detailed Description Text (139):
The PCB Address Register is loaded with an address of a Printer Control Block. Loading this register starts a Print Engine Video Controller operation. When the PCB Address Register is loaded, the Print Engine Video Controller reads the PCB parameters from memory. The PCB contains the starting address of the page image, its warp, width, and height, and other parameters and settings. The exact structure of the PCB is discussed below in section 2.3.

Detailed Description Text (140):
The PCB Address Register is "doubled buffered" to maximize Print Engine Video Controller operation. A second PCB operation to be started prior to the completion of a previous operation. The second PCB operation will automatically be started upon the completion of the first operation. The status of a PCB is indicated by the PCB loaded when the PCB EMPTY bit is set. Refer to Section 2.2.7 for a detailed discussion of the function of this status bit.

Detailed Description Text (144):
The Interrupt Event and Interrupt Mask registers are discussed in detail below, section 2.4. They indicate the source of an interrupt, and allow masking of each interrupt source. There is a total of eight interrupt sources.

Detailed Description Text (147):
The Graphics Execution Unit and Print Engine Video Controller Status Register (GEU/PVC) is a 4 bit register that indicates the state of the Graphics Execution Unit and Print Engine Video Controller. These bits indicate when the respective controllers are operating and their memory control structures are available for use.

Detailed Description Text (149):
The DLA EMPTY status bit indicates the state The second load of the Display List Address Register will cause DLA EMPTY to be reset, and this bit will stay reset until the Graphics Execution Unit has finished executing the first display list and started on the second. If a display list address is loaded when DLA EMPTY is not set, the address is ignored and lost. The PRINT BUSY bit indicates that the Print Engine Video Controller is in operation. The Print Engine Video Controller sets PRINT BUSY true when the PCB Address Register is loaded. This bit will be reset upon the completion of a PCB operation. A PCB operation is complete when the last video data word is fetched from memory and loaded into the internal PBC video FIFO. The PRINT BUSY bit will not be reset upon the completion of a PCB operation if a subsequent PCB address has been loaded into the PCB Address Register prior to the

completion of the previous PCB operation.

Detailed Description Text (154):
The Page Image Bit Address is a pointer to the starting address of the page image or band. It is interpreted as bit address, and must be byte aligned (the PBC zeroes the three least significant bits). PBC has the capability to interface to a variety of print engines. The address written into the Page Image Address Register will depend on the scanning direction of the print engine. Table 1 indicates the proper starting address for each of the 4 scanning options supported by PBC.

Detailed Description Text (160):
The Video parameter values are sampled at the top of every page. In banding applications, the parameters are read on the first band of the page, and ignored for successive bands in the page.

Detailed Description Text (162):
The Control field of the PCB contains three control bits. LAST BAND is used by software to indicate the final band of the page; LAST BAND must also be set for unbanded page images. The Print Engine Video Controller returns an "page end" interrupt when the last word of the last scanline of the page image has been read from memory. Software can then reclaim page image memory space.

Detailed Description Text (163):
LAST BAND also determines the print engine signals used for video synchronization for the next band to be printed. When LAST BAND is reset, the Print Engine Video Controller will start the transfer of video data for the next band on the first assertion of LSYNC. When LAST BAND is set, the Video Controller will wait for the assertion of FSYNC and then will start the data transfer for the next band on the next LSYNC.

Detailed Description Text (170):
The Graphics Execution Unit generates an interrupt when it becomes idle upon completion of the execution of a display list and the Display List Address register is empty. If the Graphics Execution Unit detected an error during the execution of a display list, both BLT DONE and BLT ERROR are set. The Graphics Execution Unit must be reset after a BLT ERROR is returned (see section 2.5, below).

Detailed Description Text (171):
The Print Engine Video Controller has two sources for interrupts: BAND BEGIN and PAGE END. BAND BEGIN is generated at the start of a page or band. The exact timing of the BAND BEGIN interrupt is dependent on the position within the printed page. If the Print Engine Video Controller is waiting for an FSYNC at the top of a page, then BAND BEGIN is generated immediately after FSYNC is received from the print engine. If the page is being rendered in bands and the Print Engine Video Controller is on a band further down the page, then BAND BEGIN is generated immediately after the last parameter of the PCB is read from memory.

Detailed Description Text (172):
PAGE END is generated in response to a PCB with LAST BAND set immediately after the last word of the last scanline is read from memory.

Detailed Description Text (173):
Three additional bits are present in the upper half of the Interrupt Event register: VIDEO UNDER, BAND UNDER, and PRINT ERROR. These bits are status bits, not interrupt sources. These bits are valid when the BAND BEGIN or PAGE END bits are asserted. A full explanation of these bits is provided in the following section.

Detailed Description Text (178):
The three types of errors that can be detected by the Print Engine Video Controller are VIDEO UNDER, BAND UNDER, and PRINT ERROR.

Detailed Description Text (179):
VIDEO UNDER indicates that a video underrun occurred during the previous band (or page). This error occurs if the print engine requests video data faster than the Video Controller can access memory.

Detailed Description Text (180):
BAND UNDER indicates a banding error. This error occurs when the Print Engine Video Controller is between bands, and an LSYNC was received from the print engine (1) before another PCB address was received, or (2) while the Print Engine Video Controller is busy reading the PCB parameters or the first words of the page image. Band underruns usually point to a page description that is too complicated for the given banding environment.

Detailed Description Text (188):
When the destination bitmap is a banded bitmap, the transfer terminates prematurely if the frame extends below the bottom of the band. This is considered a band fault. (The transfer will resume when the display list is rerun to render the next band of page image.)

Detailed Description Text (190):
A scanline transfer involves operating on a specified set of scanline runs on one or more specified bitmaps. The set of scanline runs is defined by a scanline table, which contains a series of bitstring specifiers. Each bitstring specifier is a compressed run-length encoding of scanline run. The compressed format of scanline tables not only save memory, but also improve performance since less memory fetches have to be performed.

Detailed Description Text (192):
When the destination bitmap is a banded bitmap, the transfer terminates prematurely if the scanline frame extends below the bottom of the band. This is considered a band fatfit. (Execution of the scanline table will resume when the display list is rerun to render the next band of page image.)

Detailed Description Text (200):
These graphic orders rely on parameters being previously set with certain initialization orders. Removing parameters that are typically common to several bitBLT operations makes each bitBLT graphic order smaller and results in a display list which occupies less memory space.

Detailed Description Text (203):
As with the bitBLT orders, the scanline graphic orders rely on parameters being previously set with certain initialization orders. Removing parameters that are typically common to several scanline operations makes each scanline graphic order smaller and results in a display list which occupies less memory space.

Detailed Description Text (211):
Operations which involve transferring pixels to frames are particularly useful when a small rectangle of a larger image must be cut and saved for future reference. The frame is the smallest possible storage means for the small rectangle because the scanlines are completely compacted in memory; the end of one scanline and the beginning of the next have no unused bits between them.

Detailed Description Text (213):
An unbanded bitmap is simply a bitmap which has no logical coordinate associated with its Y axis. When specifying an unbanded bitmap in a transfer of data, it is assumed that the bitmap is of sufficient size to complete the transfer. This requires that sufficient memory be allocated to contain the transfer. Another way of thinking about it is that no bounds checks are made for unbanded bitmaps.

Detailed Description Text (215):
A banded bitmap is simply a bitmap which has a logical coordinate associated with its Y axis. When specifying a banded bitmap in a transfer of data, the transfer of data is checked against the Y bounds of the bitmap. If the data would fall beyond the Y bounds of the bitmap, a band fault occurs, and the transfer of data is prematurely terminated.

Detailed Description Text (216):
Upon termination of a graphic order, all parameters necessary to resume execution at the point where the band fault occurred are written back to the original graphic

order, and the graphic order's band number is incremented (or decremented) to reflect the band where execution of the graphic order should resume.

Detailed Description Text (217):
By automatically updating graphic orders which generate band faults, graphic orders which span multiple bands can be executed without the need for software intervention.

Detailed Description Text (249):
Each scanline graphic order specifies a pointer to a scanline table. The scanline table contains a list of bitstring specifiers that specify the individual scanline runs of an image. Each scanline table starts and ends with a 0.times.0000 short word terminators. Note that the null terminators can be shared between scanline tables placed adjacent in memory. That is, the ending 0.times.0000 of one scanline table can serve as the starting 0.times.0000 for a subsequent scanline table. The dual termination of the scanline table structure is required for duplex printing, when the PBC reads the scanline table in both forward and reverse directions.

Detailed Description Text (252):
One of the major benefits of images represented as a series of scanlines is a reduction in the memory storage required to represent the image. This is further augmented by supporting different sized bitstring specifier formats that can economize on short displacements and runs, or allow any pixel in a bitmap to be reached with a single specifier.

Detailed Description Text (253):
PBC supports three bitstring specifiers: a 16-bit specifier for small displacements and runs, a 32-bit specifier for medium displacements and runs, and a 48-bit specifier for large displacements and runs. The specifiers are designed to be recognizable whether the scanline table is read from memory in a forward or backward direction (more precisely, the ID bits in the 32- and 48-bit specifiers are duplicated to allow parsing of the scanline table in either direction). This is required to properly handle scanline graphic orders in a duplex printing environment.

Detailed Description Text (256):
All of the bitstring specifiers are multiples of 16-bits and must always be located on short word memory boundaries (0 mod 2 byte addresses).

Detailed Description Text (272):
As previously stated, a halftone bitmap is automatically replicated in both the horizontal and vertical directions. This is fairy easy to automate for the bitBLT graphic orders since the area to be operated on is a rectangle. For scanline graphic orders, however, the task is rather complex since each scanline run describes an individual area to be operated on. For the 16-bit bitstring specifier, the task is relatively straight forward and can be carded out with a minimal number of overhead cycles. For the 32- and 48-bit specifiers, the task is considerably more complex because of the much larger displacement values associated with them. The task can be carded out, but at significant number of overhead cycles. To reduce the overhead cycle penalty for the larger bitstring specifier formats, a companion halftone table is employed. This eliminates virtually all overhead cycles in return for slightly higher memory storage requirements for most typical scanline tables.

Detailed Description Text (299):
When executing scanline graphic orders to a banded bitmap, PBC performs boundary checking to detect a band fault before executing each bitstring specifier. It does not, however, check for a band fault while processing the run length of pixels. There are two usages of the bitstring specifiers during banding applications which can cause unwanted destruction of data. The destruction arises when the specifier causes processing to occur outside of the banded bitmap.

Detailed Description Text (300):
The first case occurs when a bitstring specifier contains a signed offset which references a previous band (refer to FIG. 20A-20D). The PBC checks only the lower bound of the banded bitmap, so the violation is undetected. The PBC will process the

bitstring specifier run at a memory location which is not contained within the target bitmap.

Detailed Description Text (303):
This section presents each of the PBC's 25 graphic orders in alphabetical order. For each graphic order, a functional description, opcode, operand format, and definitions of its operands are given. In addition, at the end of this section, two summary reference tables (Tables 26 and 27) are presented.

Detailed Description Text (306):
The addressing conventions discussed in section 1.6 above also apply to graphic order address parameters. All address fields are equally 32 bits wide. Address parameters that reference bitmaps specify 32-bit bit addresses, while all other others require conventional 29-bit byte addresses.

Detailed Description Text (311):
1. Bit address: A 32-bit bit address which can point to any bit location in memory; used to specify bitmap locations and bitmap pixels.

Detailed Description Text (312):
2. Bit address, byte aligned: A 32-bit bit address with its least three significant bits set to zero; points to a bit location in memory that is the first bit in a byte. Used to specify bitmap locations.

Detailed Description Text (313):
3. Byte address: A 29-bit byte address which can point to any byte location in memory; used for referencing graphic order instructions.

Detailed Description Text (314):
4. Byte address, short word aligned: A 29-bit byte address with its least significant bit set to zero; points to a byte in memory that is the first byte in a short word. Used to address scanline tables and halftone companion tables.

Detailed Description Text (316):
For graphic orders that render images to a frame or unbanded bitmaps, address parameters are interpreted as physical addresses, and the PBC uses them directly to access memory. In the case of banded bitmaps, where only a portion of the physical page's image is present in memory, address parameters are interpreted as logical addresses.

Detailed Description Text (317):
Logical addresses must be translated by the PBC to physical space before execution of the graphic order begins. Translation information is provided to the PBC at the same time the banded bitmap dimensions are defined (via the SET.sub.-- BBMAP order). Physical addresses are translated back to logical addresses when band faults occur and require parameters to be written back to the graphic order.

Detailed Description Text (322):
5.2 BAND NUMBERS

Detailed Description Text (323):
Each of the PBC graphic orders which affect the bands of the image contain a band number. The band number is used by the PBC to determine when to execute the order. Band numbers increase in value from the top band to the bottom band of the image.

Detailed Description Text (324):
The band number for a graphic order must be determined by software. It must be the value of the band where the first scanline of a graphic order resides when printing the front side of a page, or the value of the band where the last scanline of a graphic order resides when printing the back side of a page (duplex printing).

Detailed Description Text (325):
The PBC automatically adjusts the band number when the graphic order spans multiple bands. It increments the band number when printing the front side of a page and decrements the band number when printing the back side of a page.

Detailed Description Text (330):
When a band fault is detected, the PBC rewrites the graphic order to update some of its parameters. The BAND number is incremented (or decremented, for backside printing). DA is repositioned to the starting pixel of each respective frame to be processed in the next band. Last, FH and HYR are written back with the number of remaining scanlines in their respective frames to be transferred.

Detailed Description Text (334):
When a band fault is detected, the PBC rewrites the graphic order to update some of its parameters. The BAND number is incremented (or decremented, for backside printing). DA and SA are repositioned to the starting pixel of each respective frame to be processed in the next band. Last, FH is written back with the number of remaining scanlines in the bitBLT frame to be transferred.

Detailed Description Text (340):
When a band fault is detected, the PBC rewrites the graphic order to update some of its parameters. The BAND number is incremented (or decremented, for backside printing). DA, SA, and HA are repositioned to the starting pixel of each respective frame to be processed in the next band. Last, FH and HYR are written back with the number of remaining scanlines in their respective frames to be transferred.

Detailed Description Text (363):
The SET.sub.-- B B MAP graphic order specifies the structure of a banded bitmap. The current band number, render direction, warp, and physical base address of the bitmap are given, along with the logical address of the first bit inside, and outside, the bitmap. These parameters are used in all subsequent graphic orders which operate on a banded bitmap.

Detailed Description Text (365):
The SET.sub.-- BBMAP graphic order specifies a current band number; the current band number is used to compare against the band numbers found in subsequent bitBLT and scanline graphic orders. The result of each band number comparison determines whether a graphic order is executed during the current pass of the display list. Remember that a "banded" display list is executed several times, one pass for each band of the entire page.

Detailed Description Text (366):
The DUPLEX byte contains a one-bit flag to indicate render direction. When the least significant bit of the DUPLEX byte is set, all subsequent graphic or scanline orders that operate on banded bitmaps are to be rendered in a bottom to top fashion. In conjunction, a bottom to top render direction causes the PBC to assume that all starting pixel addresses in subsequent banding graphic and scanline orders point to the bottom or end of their respective operands instead of the top. (Orders that operate on frames or unbanded bitmaps are unaffected by the render direction, and are always rendered top to bottom.) By rendering in a reverse direction, the bands of a page can be created in opposite order to print the backside of duplex pages.

Detailed Description Text (368):
As illustrated in FIG. 35A-35C, the three address parameters SOB PA, SOB A, and EOBPA vary depending on the side of the page (front or backside) to be generated. The parameters are referenced from the top of the band when printing the front side of a page, and from the bottom of the band when printing the back side of a page.

Detailed Description Text (369):
Finally, as a banded bitBLT or scanline operation proceeds, its frame logical address is compared with the logical address of the first bit outside of the banded bitmap. The frame logical address is modified by the banded bitmap warp after each scanline is completed. When the frame logical address becomes greater than or equal to the logical address of the first bit outside of the banded bitmap, a band fault occurs. The operation is prematurely terminated and the appropriate parameters are saved for later execution on the next band.

Detailed Description Text (370):
The valid one and two operand boolean operations are listed in sections 3.9.1 and

3.9.2. A generalized algorithm for generating the mask values is given in section 3.9.4, Determining Boolean Code.

Detailed Description Text (377):
Note that the halftone bitmap width can be any value from 1 to 65,535 bits. While the PBC correctly handles widths less than 32 bits, it is strongly recommended that all halftone bitmaps be a minimum of 32 bits in order to minimize the cost of replicating small halftone patterns across large objects. (Halftone bitmaps do not need to be replicated in the Y direction.) Also be aware that 32 and 64-bit wide word-aligned halftones achieve optimum performance, since the PBC can cache these size patterns internally and minimize memory accesses.

Detailed Description Text (380):
The SW parameter is typically set to a non-zero value when it references a source which is not packed in memory. This a common occurrence in bitmapped fonts stored in font cartridges. Each scanline of each character in a font begins on a word boundary (0 mod 4 byte address). This is illustrated in FIG. 37A-37B.

Detailed Description Text (401):
When a band fault is detected, the PBC rewrites the scanline graphic order to update its parameters. The BAND number is incremented (or decremented, for backside printing). DA is written back corresponding to the pixel following the last run rendered (or for backside rendering, the pixel preceding the next bitstring specifier's run), and SLTA points to the next specifier to be executed when the rest of the scanline table is rendered to the next band.

Detailed Description Text (407):
When a band fault is detected, the PBC rewrites the scanline graphic order to update most of its parameters. The BAND number is incremented (or decremented, for backside printing). DA and HA are written back corresponding to the pixel following the last run rendered (or for backside rendering, the pixel preceding the next bitstring specifier's run), and FH and HYR are written back with the number of remaining scanlines in their respective frames to be transferred. Last, SLTA and HTTA point to the next specifier to be executed when the rest of the scanline table is rendered to the next band.

Detailed Description Text (410):
The following Table 27 (sorted by opcode) identifies each graphic order and its parameters. For more detail on the meaning and use of graphic orders and their parameters, see the sections above.

Detailed Description Text (415):
The immediate blit function draws pixels in memory. Each type of blit has a set of arguments that determine the size of the image to be drawn, its location in memory, and the particular blit function.

Detailed Description Text (441):
Order construction builds stored orders for the realtime blit software for hardware to use in building the page image. Each stored order includes the information needed by the immediate blit routines and information used by the realtime blit routines such as the first page image band affected by the order.

Detailed Description Text (451):
Call Memory Manager to deallocate temporary data structure Return

Detailed Description Text (453):
The realtime blit routine begins by generating as many bands as it has available band buffers. It then delivers these initial bands to the output interface 47 in order and returns to the caller. When interrupt handler 49 determines that a band has been printed, realtime blit processing is restarted via an "extended interrupt thread" to generate the next band in the available band buffer.

Detailed Description Text (454):
Extract information from the order to define the blit required to draw the indicated pixels or the portion of them that fit in the current band

Detailed Description Text (455):
Memory Manager 43

Detailed Description Text (456):
The Memory Manager provides RAM allocation and deallocation services such that the other functions can request and release variable size blocks of memory.

Detailed Description Text (458):
The Output Interface receives arguments defining the size of the band buffer, a completion function address, left and top margin information, a count of the number of times to print the band, and flags indicating first and last band. In the non-banded case the band is a full memory image of the page and both first and last band flags are set.

Detailed Description Text (461):
Check internal information to determine whether more scanlines in the current band buffer must be delivered

Detailed Description Text (464):
FIG. 5 is a functional block diagram of the present invention adapted for driving a video display. Each of the units shown functions in identically the same manner as described above in connection with FIG. 4 except that an entire display page is processed as a single band. Output interface 47 and realtime blit processor 37 exchange bitmap image data with frame buffer 55 which then provides a raster output to the display in a conventional manner.

Detailed Description Text (466):
The apparatus illustrated in FIG. 5 differs from the graphics operations provided by current XGA display controllers which provide straight line drawing and screen-to-screen transfers in that it accesses image data in motherboard DRAM when required via bus mastership to provide both memory-to-screen and screen-to-screen memory operations. As many glyphs to be displayed are not available within the current display image, the ability to access glyphs from other memory locations is an important feature of the invention.

Detailed Description Text (467):
The apparatus expects a list of the display changes as required by the application which is being executed by the motherboard's processor to be provided either within memory directly accessible by the apparatus or available to it through the previously mentioned bus mastership mode. When operating in bus mastership mode, it is expected that a certain number of potential processor bus cycles will be consumed in accessing the requisite data. This is mitigated by two factors. First, the current processor architectures embody a cache subsystem that protects the processor from other uses of its bus subsystem. Second, the equivalent number of processor cycles is reduced by at least two thirds, the first third being the accessing of a source pattern, the second third being the accessing of the corresponding screen destination area to retrieve its existing pattern, and the third being the accessing of the corresponding screen destination area to place the resulting pattern.

Detailed Description Paragraph Table (6):
TABLE 4 BLT2BB.sub.-- D Destination Only BitBLT to Banded Bitmap 0x30 BLT2BB.sub.-- D opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (29 bits) FW Frame width in bits (16 bits) * FH Frame height in scanlines (16 bits) (*) These operands are updated by the PBC when the frame crosses a band boundary.

Detailed Description Paragraph Table (7):
TABLE 5 BLT2BB.sub.-- SD Source/Destination BitBLT to Banded Bitmap 0x32 BLT2BB.sub.-- SD opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) FW Frame width in bits (16 bits) * FH Frame height in scanlines (16 bits) * SA Source physical bit address (32 bits) (*) These operands are updated by the PBC

when the frame crosses a band boundary.

Detailed Description Paragraph Table (8):
TABLE 6 BLT2BB.sub.-- SHD Source/Halftone/Destination BitBLT to Banded Bitmap

0x33 BLT2BB.sub.-- SHD opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) FW Frame width in bits (16 bits) * FH Frame height in scanlines (16 bits) * SA Source physical bit address (32 bits) HXR Halftone X remainder (16 bits) * HYR Halftone Y remainder (16 bits) * HA Halftone physical bit address of the starting pixel (32 bits) (*) These operands are updated by the PBC when the frame crosses a band boundary.

Detailed Description Paragraph Table (16):
TABLE 14 SET.sub.-- BBMAP Set Banded Bitmap Parameters

0x08 SET.sub.-- BBMAP opcode (8 bits) CUR.sub.-- BAND Current band number (8 bits) DUPLEX Render direction for duplex printing (1 of 8 bits) DWB Destination banded bitmap warp in bits (16 bits) SOBPA Start of band physical bit address, byte aligned (32 bits) SOBA Start of band logical bit address (32 bits) EOBPA End of band physical bit address (32 bits)

Detailed Description Paragraph Table (25):
TABLE 23 SL2BB.sub.-- D Destination Only Scanline Transfer to Banded Bitmap

0 .times. 34 SL2BB.sub.-- D opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) * SLTA Scanline table physical byte address, short word aligned (29 of 32 bits)

(*) These operands are updated by the PBC when the frame crosses a band boundary.

Detailed Description Paragraph Table (26):
TABLE 24 SL2BB.sub.-- HD Halftone/Destination Scanline Transfer to Banded Bitmap

0 .times. 35 SL2BB.sub.-- HD opcode (B bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) HXR Halftone X remainder (16 bits) * HYR Halftone Y remainder (1 6 bits) * HA Halftone physical bit address of the starting pixel (32 bits) * SLTA Scanline table physical byte address, short word aligned (29 of 32 bits) * HTTA Halftone companion table physical byte address, short word aligned (29 of 32 bits)

(*) These operands are updated by the PBC when the frame crosses a band boundary.

Detailed Description Paragraph Table (29):
TABLE 27 GRAPHIC ORDER FORMATS

STOP 0 .times. 00 STOP opcode (8 bits) JUMP 0 .times. 01 JUMP opcode (8 bits) GOA Graphic order physical byte address (29 of 32 bits) SET.sub.-- BBMAP 0 .times. 08 SET.sub.-- BBMAP opcode (8 bits) CUR.sub.-- BAND Current band number (8 bits) DUPLEX Render direction for duplex printing (1 of 8 bits) DWB Destination banded bitmap warp in bits (16 bits) SOBPA Start of band physical bit address, byte aligned (32 bits) Start of band logical bit address (32 bits) SOBA EOBPA End of band physical bit address (32 bits) SET.sub.-- UBMAP 0 .times. 09 SET.sub.-- UBMAP opcode (8 bits) DWU Destination unbanded bitmap warp in bits (16 bits) SET.sub.-- SBMAP 0 .times. 0A SET.sub.-- SBMAP opcode (B bits) SW Source bitmap warp in bits (16 bits) SET.sub.-- HTBMAP 0 .times. 0B SET.sub.-- HTBMAP opcode (8 bits) HZ Halftone bitmap total size in bits (32 bits) HW Halftone bitmap width in bits (16 bits) HH Halftone bitmap height in scanlines (16 bits) SET.sub.-- BOOL.sub.-- D 0 .times. 0C SET.sub.-- BOOL.sub.-- D opcode (8 bits) BOOL.sub.-- D Destination-only boolean mask (8 bits) SET.sub.-- BOOL.sub.-- HD 0 .times. 0D SET.sub.-- BOOL.sub.-- HD Opcode (8 bits) BOOL.sub.-- HD Halftone/destination boolean mask (8 bits) SET.sub.-- BOOL.sub.-- SD 0 .times. 0E SET.sub.-- BOOL.sub.-- SD opcode (8 bits) BOOL.sub.-- SD

Source/destination boolean mask (8 bits) SET.sub.-- BOOL.sub.-- SH 0 .times. 0F SET.sub.-- BOOL.sub.-- SHD opcode (8 bits) D BOOL.sub.-- SHD Source/destination boolean mask (8 bits) BLT2F.sub.-- D 0 .times. 10 BLT2F.sub.-- D opcode (B bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) FH Frame height in scanlines (16 bits) BLT2F SD 0 .times. 12 BLT2F SD opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) FH Frame height in scanlines (16 bits) SA Source physical bit address (32 bits) BLT2F.sub.-- SHD 0 .times. 13 BLT2F.sub.-- SHD opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) FH Frame height in scanlines (16 bits) SA Source physical bit address (32 bits) HXR Halftone X remainder (16 bits) HYR Halftone Y remainder (16 bits) HA Halftone physical bit address of the starting pixel (32 bits) SL2F.sub.-- D 0 .times. 14 SL2F.sub.-- D opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) SLTA SLT physical byte address, short word aligned (29 of 32 bits) SL2F.sub.-- HD 0 .times. 15 SL2F.sub.-- HD opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) HXR Halftone X remainder (16 bits) HYR Halftone Y remainder (16 bits) HA Halftone physical bit address of the starting pixel (32 bits) SLTA SLT physical byte address, short word aligned (29 of 32 bits) HTTA Halftone companion table physical byte address, short word aligned (29 of 32 bits) BLT2UB.sub.-- D 0 .times. 20 BLT2UB.sub.-- D opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) FH Frame height in scanlines (16 bits) BLT2UB.sub.-- SD 0 .times. 22 BLT2UB.sub.-- SD opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) FH Frame height in scanlines (16 bits) SA Source physical bit address (32 bits) BLT2UB.sub.-- SHD 0 .times. 23 BLT2UB.sub.-- SHD opcode (8 bits) DA Destination physical bit address (32 bits) FW Frame width in bits (16 bits) FH Frame height in scanlines (16 bits) SA Source physical bit address (32 bits) HXR Halftone X remainder (16 bits) HYR Halftone Y remainder (16 bits) HA Halftone physical bit address of the starting pixel (32 bits) SL2UB.sub.-- D 0 .times. 24 SL2UB.sub.-- D opcode (8 bits) DA Destination physical bit address (32 bits) SLTA SLT physical byte address, short word aligned (29 of 32 bits) SL2UB.sub.-- HD 0 .times. 25 SL2UB.sub.-- HD opcode (8 bits) DA Destination physical bit address (32 bits) HXR Halftone X remainder (16 bits) HYR Halftone Y remainder (16 bits) HA Halftone physical bit address of the starting pixel (32 bits) SLTA SLT physical byte address, short word aligned (29 of 32 bits) HTTA Halftone companion table physical byte address, short word aligned (29 of 32 bits) BLT2BB.sub.-- D 0 .times. 30 BLT2BB.sub.-- D opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (29 bits) FW Frame width in bits (16 bits) * FH Frame height in scanlines (16 bits) BLT2BB.sub.-- SD 0 .times. 32 BLT2BB.sub.-- SD opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) FW Frame width in bits (16 bits) * FH Frame height in scanlines (16 bits) * SA Source physical bit address (32 bits) BLT2BB.sub.-- SHD 0 .times. 33 BLT2BB.sub.-- SHD opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) FW Frame width in bits (16 bits) * FH Frame height in scanlines (16 bits) * SA Source physical bit address (32 bits) HXR Halftone X remainder (16 bits) * HYR Halftone Y remainder (16 bits) * HA Halftone physical bit address of the starting pixel (32 bits) SL2BB.sub.-- D 0 .times. 34 SL2BB.sub.-- D opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) * SLTA Bitstring Specifier physical short word address (28 of 32 bits) SL2BB.sub.-- HD 0 .times. 35 SL2BB.sub.-- HD opcode (8 bits) * BAND Band number when graphic order is executed * DA Destination logical bit address (32 bits) HXR Halftone X remainder (16 bits) * HYR Halftone Y remainder (16 bits) * HA Halftone physical bit address of the starting pixel (32 bits) * SLTA SLT physical byte address, short word aligned (29 of 32 bits) * HTTA Halftone companion table physical byte address, short word aligned (29 of 32 bits)

(*) These operands are updated by the PBC when the frame crosses a band boundary.

Detailed Description Paragraph Table (30):

New Page The New Page call performs initialization for a new page: Determine page image memory requirements from input arguments Call the Memory Manager to allocate band buffers for the page image If Memory Manager cannot supply required memory Then if application passed NO.sub.-- PGWAIT flag or no output in progress Return with error Wait for memory to become available Initialize memory and control structure for page Return with no error End

Page End Page starts image printing if the page is not banded or starts the Realtime Blit if the image is banded. Save the address of the application completion function Set initial parameters such as current band number and copy number in the control structure for this page If the page is not banded Call Output Interface to deliver the image buffer for output Return Else Call the Realtime Blit routine, Start Next Page Return All Blits With User Destinations If the page is not banded or the application called with the DO.sub.-- ONCE flag Call Immediate Blit to perform the pixel drawing service Return Else Call Order Construction to create Stored Order for the blit Return All Blits With Page Image Destinations If the page is not banded Call Immediate Blit to perform the pixel drawing service Return Else Call order Construction to create Stored Order for the blit Return

---

Detailed Description Paragraph Table (32):

Build a stored order from the blit arguments passed If the application called with the VOLATILE flag for the source bitmap Call the memory manager to get space to copy the bitmap If there is not enough memory available Return with error code Else copy the source bitmap If the requested blit is not subject to compaction Call order compaction terminate data structure routine Else Call order compaction add order routine Return

---

Detailed Description Paragraph Table (33):

Call Memory Manager to allocate space for data structure If error Return error Initialize new data structure Place input pixel groups in the new data structure Return

---

Detailed Description Paragraph Table (35):

Start Next Page: If this page has a pre-allocated band buffer If Band Generation is busy Queue the page control structure Return Call Band Generation passing the page control structure Return Else Return Band Generation Entry Point: While image generation not done If all image band buffers are busy Return While an order in the list of Stored Orders has its band number equal to the current band number {

Detailed Description Paragraph Table (36):

Call the low level blit function If the order cannot be completed in the current band Update the band number in the order to current band + 1 Update other parameters as necessary Else Mark the order as complete If this is the first band of the page Set first band flag to pass to Output Interface Also set completion routine address If this is the last band of the page Set last band flag to pass to Output Interface Call Output Interface passing output structure If last band of image was generated If another image copy required Decrement copy count Reset band number to 1 in current control structure Else If another image is queued Get its control structure from queue Else Set band generation done } Return Completion Function: (Called by Interrupt Handler) Mark band buffer idle If more bands in current image and Band Generation is idle Start Band Generation via "extended interrupt thread" Else Call PDL interpreter completion function if there is one Return

---

Detailed Description Paragraph Table (37):

Allocate Block Search internal data structures for a free memory block equal or greater in size than the size requested If a block equal to the size requested is found Mark block as in use Return to caller with address of block Else Divide free block into two Mark the new block for the caller as in use Return to caller with address of block Free Block Verify address of block If address invalid Return with error code Mark block as free Return with no error

---

Detailed Description Paragraph Table (38):

If the output device is busy Place band information on internal queue Return Else Initialize internal data structures Start the output device Wait for synchronization signal Copy the first scanline or multiple scanlines to FIFO or start DMA device to begin image delivery Return

---

Detailed Description Paragraph Table (39):

If no more scanlines in current band Call Realtime Blit Completion Function Dequeue next band If no more bands Return Else Initialize internal information for this band Copy to FIFO or start DMA Return

---

Current US Class (1):
345

US Reference Patent Number (2):
4679038

CLAIMS:

1. An apparatus for creating an image which includes graphics information for display on a continuous synchronous raster output device, said apparatus comprising:

a) means for receiving and interpreting commands in a high level graphics language which define the image to be output;

b) means for generating a set of graphics orders from the interpreted high level graphics language commands, wherein a graphics order is a graphics function represented as a low level primitive;

wherein said set of graphics orders comprises initialization orders, flow control orders, bitmap transfer orders, and scanline orders;

c) means for generating a bitmap image from said graphics orders;

d) means for outputting said bitmap image to said output device at a speed required by the output device, said graphics orders representing a complete page image and using an amount of memory which is less than the amount of memory which would be needed to store the complete page image as a bitmap image;

wherein said graphics orders generation means comprises:

a) graphics interface subsystem means for receiving said interpreted high level graphics language commands and passing said received commands to at least one of a memory manager means, an immediate image data transfer means and an order construction means:

b) said memory manager means for allocating and deallocating blocks of a memory in response to calls from said graphics interface subsystem means, said order construction means and an order compaction means:

c) said immediate image data transfer means for drawing pixels in a user memory based upon pixel drawing commands passed by said graphics interface subsystem means:

d) said order construction means for building stored orders based upon blit arguments passed by said graphics interface subsystem means:

e) said order compaction means for combining multiple orders operating on the same destination pixel groups built by said order construction means.

2. The apparatus defined by claim 1 wherein said bitmap image generation means comprises a realtime image data transfer means for traversing said stored orders and converting the stored orders into bitmap images corresponding to sections of said image to be output.

3. The apparatus defined by claim 2 further comprising scanline table storage means for storing bitmap image data in a scanline run format and wherein said realtime

image data transfer means retrieves said scanline run formatted bitmap image data to generate at least one of said sections of said image.

5. A method for creating an image which includes graphics information for display on a continuous synchronous raster output device, said method comprising the steps of:

a) receiving and interpreting commands in a high level graphics language which define the page image to be output;

b) generating a set of graphics orders from the interpreted high level graphics language commands, wherein a graphics order is a graphics function represented as a low level primitive such that:

i) said graphics orders are processed into bands of bitmap images which are delivered to the output device at a speed required by the output device, said graphics orders representing a complete page image and using an amount of memory which is less than the amount of memory which would be needed to store the complete page image as a bitmap image; and

ii) while the graphics information in a first one of said band is being output by said output device, the bitmap images for a second one of said bands are being generated;

c) generating said bands of bitmap images from said graphic orders;

d) outputting said generated bands of bitmap images to said output device;

e) declaring a band fault if one of said graphics orders produces only a partial bitmap image within one of said bands:

f) modifying said one graphics order for processing in a next of said bands so as to produce only a remaining part of said partial bitmap image.

6. A method for creating an image which includes graphics information for display on a continuous synchronous raster output device, said method comprising the steps of:

a) receiving and interpreting commands in a high level graphics language which define the image to be output;

b) generating a set of graphics orders from the interpreted high level graphics language commands, wherein a graphics order is a graphics function represented as a low level primitive, wherein said graphics orders are processed into bands of bitmap images for delivery to the output device at a speed required by the output device, said graphics orders representing a complete page image and using an amount of memory which is less than the amount of memory which would be needed to store the complete page image as a bitmap image, wherein said set of graphics orders comprises initialization orders, flow control orders, bitmap transfer orders, and scanline orders;

c) generating said bitmap image from said graphic orders;

d) outputting said bitmap image to said output device;

e) declaring a band fault if one of said graphics orders produces only a partial bitmap image within one of said bands:

f) modifying said one graphics order for processing in a next of said bands so as to produce only a remaining part of said partial bitmap image.